МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Антикафе

Студентка гр. 0382 Морева Е.С

Студентка гр. 0383 Владыко А.О.

Студентка гр. 0383 Петровская Е.С.

Преподаватель Заславский М.М.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студенты

Морева Е.С

Группа 0382

Владыко А.О.

Петровская Е.С.

Группа 0383

Тема проекта: Антикафе

Исходные данные:

Необходимо реализовать приложение для импорта / хранения / поиска / визуализации экспериментов инструмента Sumo с помощью СУБД Neo4j.

Содержание пояснительной записки:

«Содержание», «Введение», «Сценарий использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 26.09.2023

Дата сдачи реферата: 02.02.2024

Дата защиты реферата: 02.02.2024

Студентка гр. 0382		Морева Е.С
Студентка гр. 0383		Владыко А.О.
Студентка гр. 0383		Петровская Е.С.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была сформулирована и утверждена индивидуальная тема - разработка веб-сервиса для посетителей и сотрудников антикафе с помощью СУБД MongoDB. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h23-anticafe

ANNOTATION

The course was to develop a team application for one of the identified topics. An individual theme was formulated and approved - the development of a web service for visitors and employees of the anticafe using the MongoDB DBMS was chosen. You can find the source code and all additional information at: https:/github.com/moevm/nosql2h23-anticafe

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1 Актуальность решаемой проблемы

Цель работы - создать быстрое и удобное веб-приложения для антикафе.

1.2 Постановка задачи

Сервис должен предоставлять функционал для решения следующих задач:

- Позволять пользователям-посетителям регистрироваться, бронировать столики и заказывать еду
- Предоставлять пользователям-работникам антикафе инструменты для работы с гостями и базой данных

1.3 Предлагаемое решение

Для решения поставленной задачи необходимо разработать web приложение для антикафе.

1.4 Качественные требования к решению

Требуется разработать веб-приложение с использованием СУБД MongoDB, фреймворка Vue.js.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

1. Главный экран (Рис. 1)

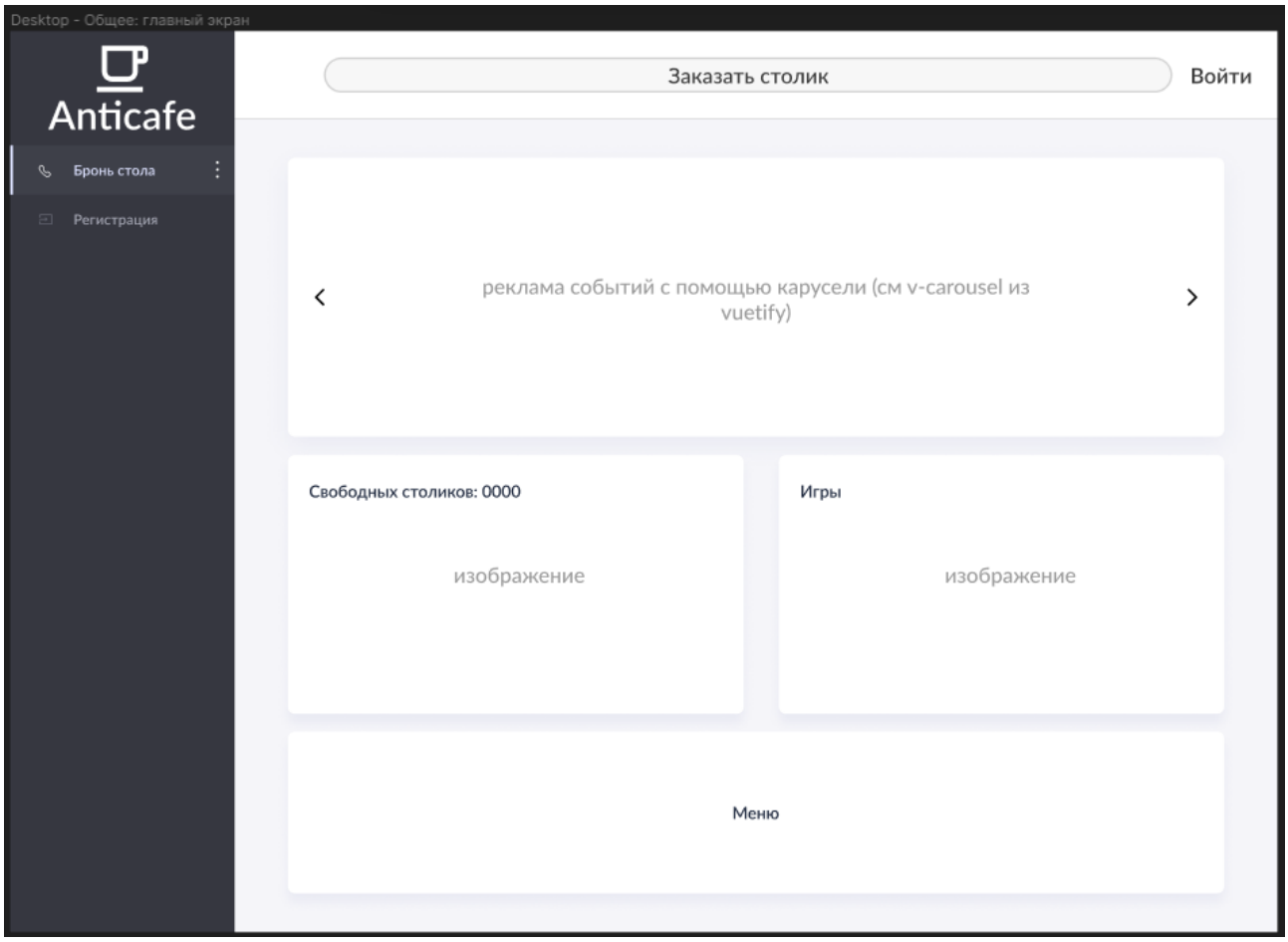

Рисунок 1 - главный экран

2. Экран входа (Рис. 2)

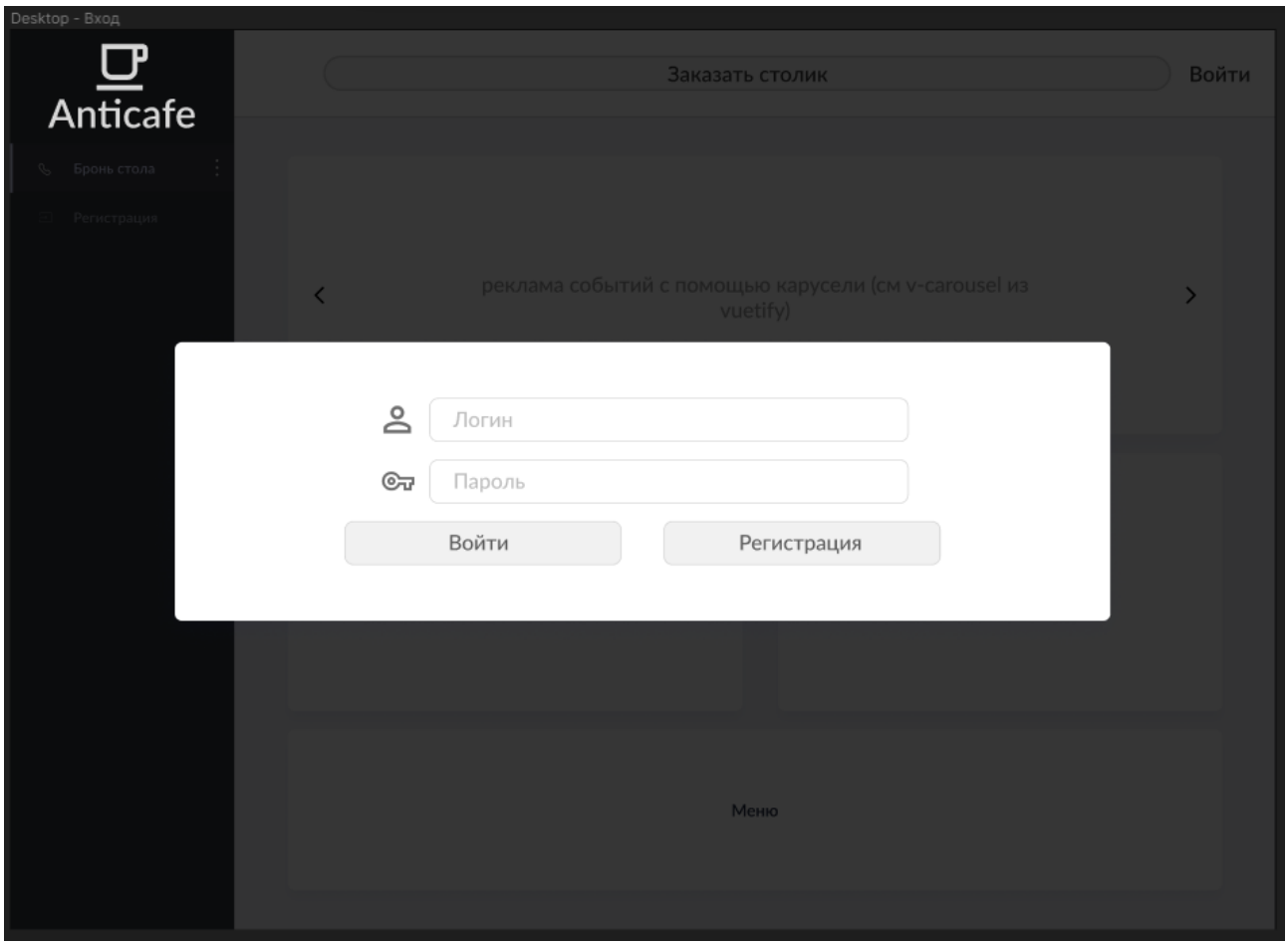

Рисунок 2 - Экран входа

3. Экран заказа стола (Рис. 3)

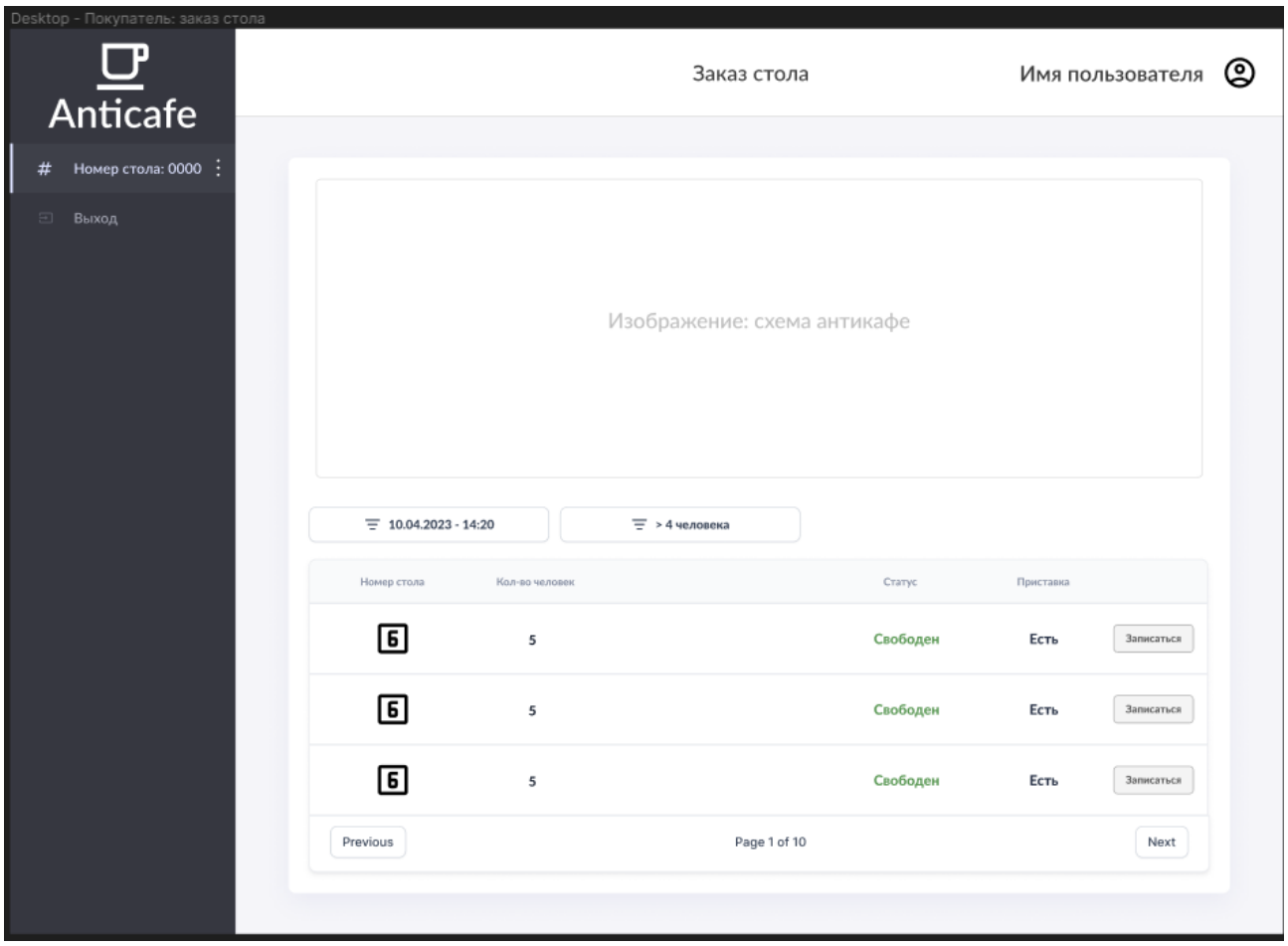

Рисунок 3 - Экран заказа стола

4. Страница главного экрана работника (Рис. 4)

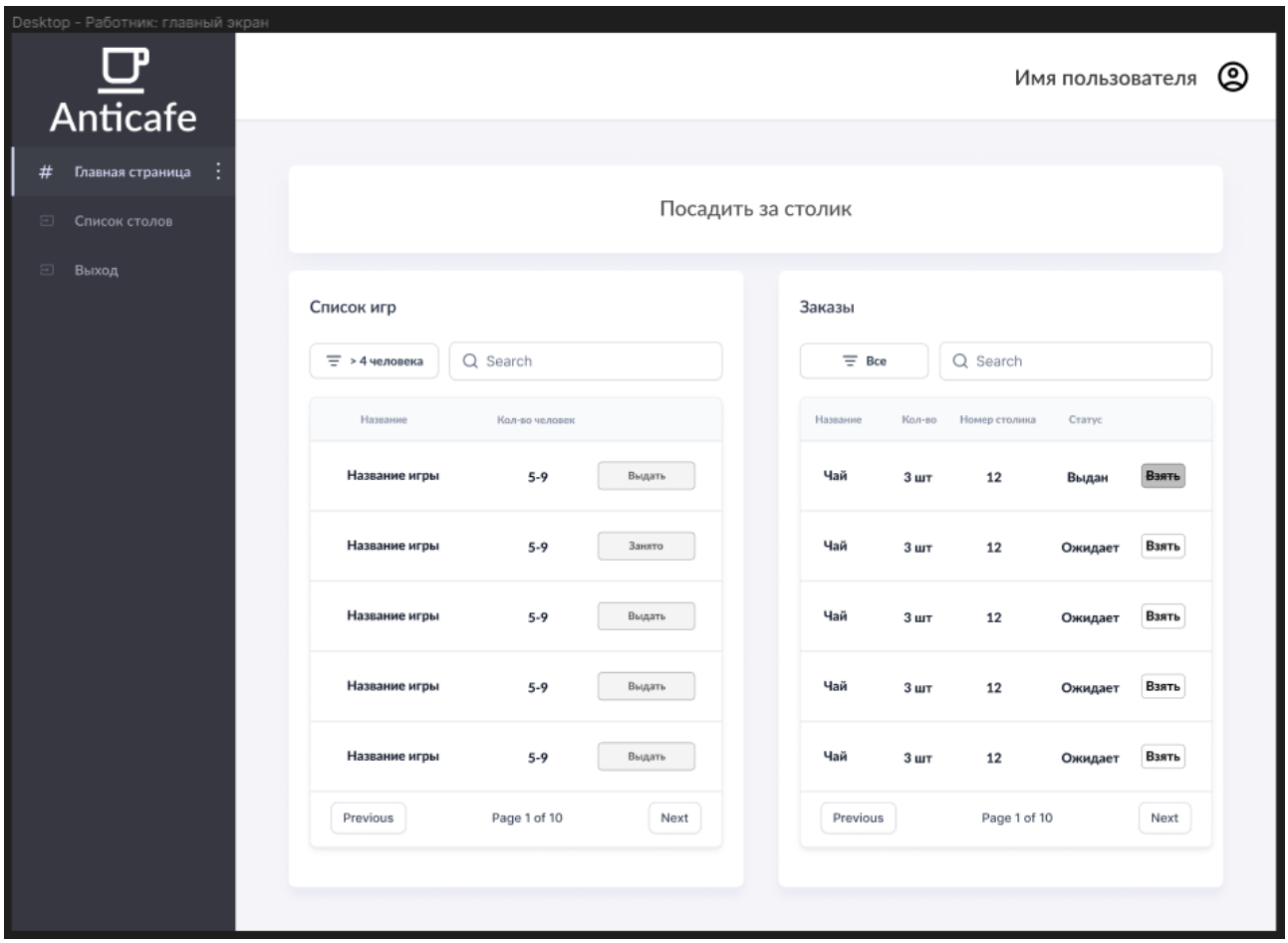

Рисунок 4 - Страница главного экрана работника

2.2. Сценарии использования для задачи

2.2.1. Действующее лицо: Пользователь

- Основной сценарий:

1. Пользователь заходит на главную страницу и видит рекламу акций и предстоящих событий в антикафе
2. Пользователь нажимает «заказать столик» или «вход/регистрация»
3. Его переносит на страницу авторизации/регистрации
4. Пользователь выбирает нужный столик и время
5. Столик заказан!
6. При входе в аккаунт теперь пользователь видит страницу своего столика

- Альтернативный сценарий:

1. Пользователь зашел на главную
2. Пользователь покинул сайт

2.2.2. Действующее лицо: Работник заведения

- Основной сценарий:

1. Работник заходит на главную и нажимает «вход»
2. Попадает на свою страницу и выбирает одну из возможностей «посадить посетителя за столик», «принести заказ», «выдать игру»
3. При "посадить за столик" на экране появляется окно для ввода логина посетителя

3.3 После ввода логина Работник попадает на страницу выбора столиков с таблицей

4. При выдаче игры Посетителю Работник нажимает кнопку "выдать"

4.1 На экране появляется окно для ввода логина посетителя

4.2 Работник возвращается на главный экран, игра выдана

5. Выбирая в списке заказов заказ на выдачу, Работник помечает блюдо как "в процессе"

5.1 После выдачи заказа заказчику, Работник помечает блюдо как "выдано"

2.2.3. Действующее лицо: Администратор заведения

- Основной сценарий:

1. Администратор заходит на главную и нажимает «вход»
2. Выбирает одну из возможностей «посадить посетителя за столик», «таблица заказов», «таблица игр», «таблица столиков», «таблица работников», «выгрузка/загрузка БД», «меню»
3. В каждом пункте администратор может добавить/удалить/изменить статус (игры, работника, столика, пункта меню соответственно)
4. В случае необходимости выгрузки данных из базы, администратор может нажать на соответствующую кнопку на странице администратора и применить необходимые фильтры чтобы получить более конкретные данные

5. Для загрузки данных администратор может нажать на соответствующую кнопку на странице администратора и выбрать источник бд, далее нажать "Окей" и данные будут загружены.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

3.1.1. Графическое представление данных

Графическое представление нереляционной базы данных представлено на рис. 5

Рисунок 5 - Схема нереляционной бд

3.1.2. Описание назначений коллекций, типов данных и сущностей

Описание коллекций

Guests - Посетитель - хранит Посетителей антикафе

Employees - Сотрудники - хранит Сотрудников антикафе, как Администраторов, так и обычных работников

Permissions - Разрешения - список доступных ролей-разрешений для Сотрудников

Tables - Столы - список доступных в антикафе столов

Bookings - Брони - список броней, сделанных пользователями
Games - Игры - список доступных в антикафе игр
Foods - Блюда - доступные в антикафе Блюда
Tariffs - Тарифы - возможные Тарифы в антикафе с разными ценниками и списками доступных Блюд
Orders - Заказы - активные Заказы, доступные для выполнения Сотрудниками

Описание сущностей

Guest - Посетитель: Сущность для хранения данных пользователя-посетителя.

ID каждого посетителя является ObjectID, оно же - первичный ключ.
Login и Password хранят в себе строковые значения логина и пароля, по которым любой пользователь, включая Сотрудник, входит в систему.
First_name и Second_name хранят имя и фамилию пользователя.
Contacts хранит контакты пользователя - email и phone
Bookings_id - список внешних ключей на брони из коллекции Bookings
Current_reservation - объект, описывающий детали нынешней резервации, не пуст если Посетитель на данный момент посажен за стол
Table_id - внешний ключ = номер стола
Total_cost - итоговый счет, значение высчитывается с помощью значений из Тарифа и времени, которое Посетитель провел за столом.
Time_start - время, когда Посетитель сел за стол.
Time_end - время, когда пользователь покинул Стол, при заполнении данного поля производится расчет Total_cost.

Employee - Сотрудник: Сущность для хранения данных пользователя-сотрудника.
ID - ObjectID и первичный ключ для Сотрудника
Login & Password - аналогично пользователю-посетителю
First_name & Second_name & Contacts - аналогично см. выше
Employment date - дата устройства на работу
Status - статус сотрудника (Работает/Не работает/Уволен)
Permissions_id - список внешних ключей Разрешений, по ним определяется, является ли Сотрудник также администратором

Permission - Разрешение: Сущность для создания ролей и разрешений среди Сотрудников.
ID - Первичный ключ
Name - название роли/Разрешения

Table - Стол: Сущность для хранения данных о столиках в антикафе.
ID - Первичный ключ - ObjectID
Number - номер стола

Capacity - сколько Посетителей помещается за стол
Bookings - список объектов для хранения дат и времени брони стола, существует отдельно от сущности Бронь для облегчения доступа к данным.
Date - объект-дата
Time - список объектов времени брони в конкретную дату
Guests_id - список внешних ключей Посетителей, пополняется после того, как посетители посажены за стол.
Dur_end_est - число - продолжительность брони в минутах

Booking - Бронь: Сущность связующая Столы и Посетителей, для удобства вывода на странице Посетителя созданных им Броней.
ID - первичный ключ - ObjectID
Table_id - внешний ключ, обозначение номера стола
Date - дата брони
Time - время брони
Tariff_id - внешний ключ конкретного тарифа

Tariff - Тариф: Сущность Тарифа для подсчета оплаты и списка доступных пользователю Блюд.
ID - первичный ключ - ObjectID
Name - название тарифа
Cost_per_min - стоимость минуты, проведенной за столом с данным тарифом
Foods_id - список внешних ключей блюд, включенных в данный тариф

Order - Заказ: Сущность, хранящая данные о заказах, создается, когда Посетитель заказывает доступное ему по Тарифу Блюдо из списка
ID - первичный ключ - ObjectID
Food_id - внешний ключ - обозначает вид Блюда, которое требуется принести к Столу
Amount - число Блюд в заказе
Status_done - статус заказа, если выполнен - True, иначе False
Employee_id - внешний ключ - обозначает Сотрудника, взявшего на себя выполнение данного Заказа
Table_id - внешний ключ - номер стола, к которому нужно нести заказ
Food - Блюдо: Блюдо описывает доступную в антикафе еду и напитки.

ID - первичный ключ - ObjectID
Name - название блюда
Content - строка, описывающая состав Блюда
In_stock - число доступных объектов данного Блюда на складе

Game - Игра: Сущность, описывающая доступные Игры в антикафе и учитывающая их статус.
ID - первичный ключ - ObjectID

Name - название игры - строка
Players_min - минимальное число игроков
Players_max - максимальное число игроков
Table_id - внешний ключ стола, на котором находится игра. Если игра свободна для брони, данное поле пусто

3.1.3. Оценка удельного объема информации, хранимой в модели

1. Так как объем данных типа String зависит от длины самой строки, для данного расчета положим, что всякая строка имеет длину в 6 символов в минимальном случае.

2. В базовом случае в приложении существует лишь 2 роли для Сотрудников - администратор и обычный работник, число возможных Разрешений составляет максимум 1, следовательно наибольшее число элементов в списке Разрешений у одного Сотрудника равно 1.

3. Сущность Стол хранит в себе список объектов дат под названием bookings, который может быть пуст или содержать число объектов $\leq \mathbf{max(DaysInMonth)}$. Также каждое поле даты есть список объектов-времени резервации в данную дату, число которых может быть $\mathbf{0 \leq N \leq 24}$, так как положим наименьшая длительность резервации равна 1 часу. Число списка гостей может находиться в диапазоне $\mathbf{0 \leq N \leq capacity.value()}$ конкретного Стола.

4. Сущность Посетителя хранит в себе список броней, сделанных им, и он может быть как пуст, так и иметь неопределенное число записей $\mathbf{0 \leq Bookings.length()}$, где Bookings - коллекция Броней. При этом когда Посетитель посажен за стол, его поля current_reservation заполняются соответственно.

5. Тариф хранит в себе список Блюд, которые доступны при бронировании с данным тарифом для заказа. Их число может быть $\mathbf{0 \leq Foods.length()}$, где Foods - коллекция Блюд.

Таким образом, воспользовавшись значениями согласно спецификации MongoDB составим таблицу типов данных для сущностей:

Guest	Employee	Permission	Table	Booking	Game	Food	Order	Tariff	Размер/ед.	
ObjectID	1 (+B*)	3 (+ 0-1)	1	1 (+N*)	3	2	1	4	1 (+F*)	12 bytes
Int32	0	0	0	2 (+N*)	0	2	1	1	0	4 bytes

Guest	Employee	Permission	Table	Booking	Game	Food	Order	Tariff	Размер/ед.	
Double	1	0	0	0	0	0	0	0	1	8 bytes
String	6	7	1	D*	1	1	2	0	1	1 byte per symbol + 1
Date	2	1	0	0	1	0	0	1	0	8 bytes
Boolean	0	0	0	0	0	0	0	1	0	2 bytes

$N* = [0; 31 \times 24 \times capacity.value()] \in \mathbb{N}$ - п.3

D* = $[0; 31 \times 24]$ - число возможных объектов времени

B* = $[0; Bookings.length()] \in \mathbb{N}$ - п.4

F* = $[0; Foods.length()] \in \mathbb{N}$ - п.5

Выведем формулы объемов для коллекций приведенных сущностей:

$$V_{guests} = guests.length() * (34 + 12B^* + 6strings.length())$$

$$V_{employees} = employees.length() * (51 + 7strings.length())$$

$$V_{employee-admin} = (51 + 7strings.length()) + 12admins.count()$$

- 1 Работник-администратор

$$V_{tables} = tables.length()(23 + 5D^* + 16N^*)$$

$$V_{games} = games.length()(33 + strings.length())$$

$$V_{bookings} = bookings.length()(45 + strings.length())$$

$$V_{foods} = foods.length()(18 + 2strings.length())$$

$$V_{orders} = 62orders.length()$$

$$V_{tariffs} = tariffs.length()(10 + 12F^* + strings.length())$$

Для расчета среднего размера базы данных после работы антикафе в течение одного месяца положим, что каждый посетитель в посещает заведение 2 раза в компании из 3х человек - т.е. на 1го из трех Guest придется по 2 записи в коллекции Bookings, пусть N* для каждого стола составляет 6. На каждый

Booking приходится 5 Orders. Тарифов существует 2, в одном 5 видов блюд, в другом 10. Также :

tables.length() = 10 - число столов

guests.length() = 30 - число пользователей-посетителей

employees.length() = 3 - число работников, администратор 1 из них

games.length() = 20 - число игр

foods.length() = 10 - число видов блюд

tariffs.length() = 2 - число тарифов

strings.length() = 10 - средний размер любой строки

Тогда:

V_guests = 30 * (34 + 12 * 20 + 60) = 10,02 кБ

V_employees = 3 * (51 + 7 * 10) + 12 = 0,375$ кБ

V_tables = 10(23 + 16 * 6 + 5 * 2) = 1,29$ кБ

V_bookings = 20(45 + 10) = 1,1 кБ

V_games = 20(33 +10) = 0,86 кБ

V_foods = 10(18 + 2 * 10) = 0,38 кБ

V_orders = 62*20 = 1,24 кБ

V_tariffs = 2(10 + 12 * 15 + 10) = 0,4 кБ

Тогда общий объем базы данных за месяц при данных условиях:

V = V_guests + V_employees + V_employee-admin + V_tables + V_bookings + V_games + V_foods + V_orders + V_tariffs = 10,02 + 0,375 + 1,29 + 1,1 + 0,86 + 0,38 + 1,24 + 0,4 = 15,5 кБ

Объем базы в зависимости от количества пользователей при данных условиях:

V(guests.length()) = 5,48 + 0.34 guests.length() кБ

Для подсчета минимального размера при ненулевых значениях полей сущностей базы данных положим, что в каждой коллекции есть по одному объекту, один Сотрудник и есть администратор, Стол имеет одну бронь с

одним посетителем, Посетитель посажен за стол, а все строки имеют длину 6. Тогда размер коллекций в байтах будет составлять:

Guests	Employees	Permissions	Tables	Bookings	Games	Foods	Orders	Tariffs	Сумма
Итого (bytes)	90	105	19	35	51	39	30	62	39	**470**

3.1.4. Запросы к модели, с помощью которых реализуются сценарии использования

Текст запросов:

Поиск пользователя в коллекции Guests

```
currGuest = db.Guests.find ( {login : 'login'}, {password: 'password'} );
```

Поиск свободных столов на дату 2023-09-20

```
db.Tables.find({ $or: [ { "2023-09-20": { $exists: false} },  $and: [{ "2023-09-20": { $exists: true, } }, { {$sum: $"2023-09-20".dur_end_est}: { $lt: 1440 } }])
```

Заказ столика на дату 2023-09-20 8:00

```
newBooking = db.Bookings.insertOne(
        {_id:  ObjectID()
        table_id  : selectedTableID,
        date  : selectedDate,
        time:"8:00",
        tariff_id: selectedTariffID
        }
)
currGuest.update({_id: currGuest._id }, {$push: { bookings_id: newBooking._id }})
```

3.2. Реляционная модель данных

3.2.1. Графическое представление данных

Графическое представление реляционной базы данных представлено на рис. 6

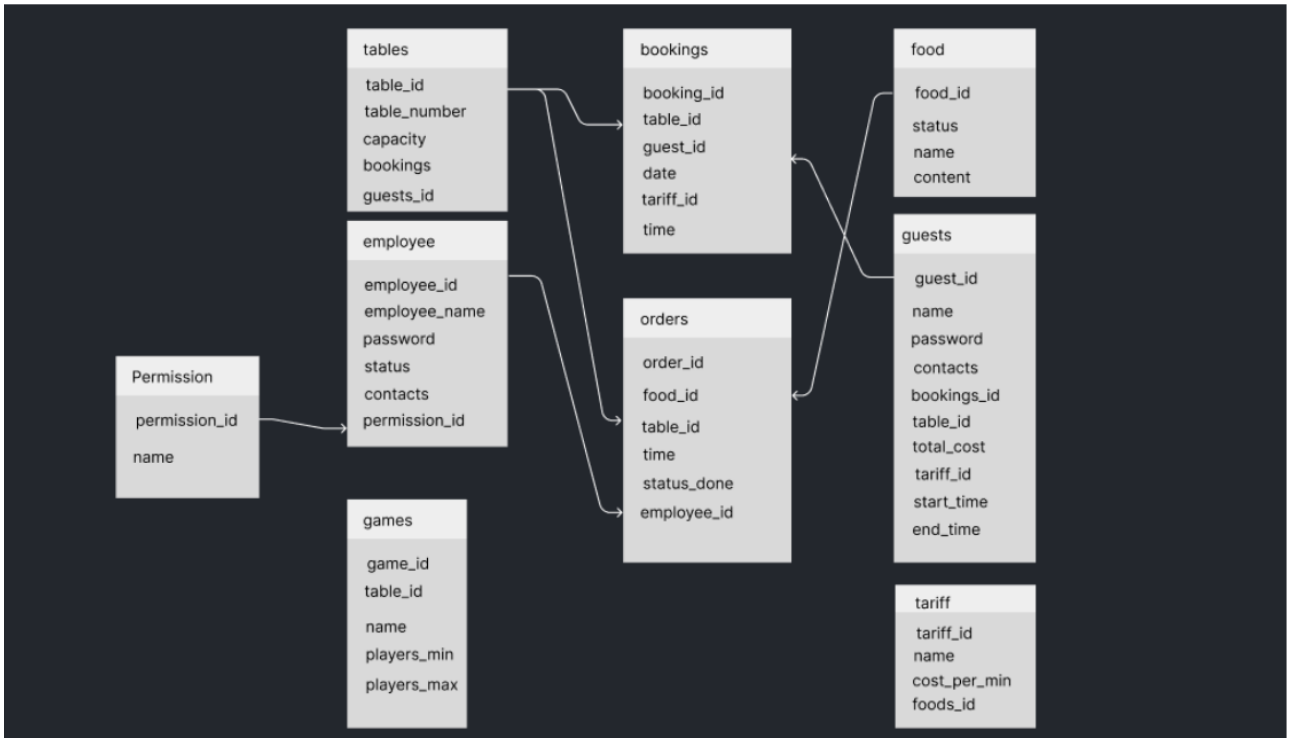

Рисунок 6 - графическое представление реляционной базы данных

3.2.2. Описание назначений коллекций, типов данных и сущностей

Таблица “tables”

table_id - уникальный идентификатор стола. Тип int16. V = 2b

table_number - номер стола в зале. Тип int16. V = 2b

capacity - сколько стол вмещает людей. Тип int16. V = 2b

bookings - когда стол забронирован. Тип String. V = n * s, где s = 14 сообщение о дате и времени, n = 4 - среднее количество броней одного стола в день. V = 56b

guests_id - уникальные идентификаторы гостей, сидящих за столом в настоящее время. Тип String. V = n * s, где s = 4 - объем для id одного пользователя, n = 4 - среднее количество человек за одним столом. V = 32b

Таблица “guests”

guest_id - уникальный идентификатор гостя. Тип int32. V = 4b

password - пароль от аккаунта. Тип int32. V = 4b

contacts - электронная почта и номер телефона. Тип String. V = 50b

bookings_id - идентификаторы броней столов, сделанных этим пользователем. Тип String. V = n * s. n ~ 1 среднее число заказанных столов. s = 4b. V = 4b

table_id - идентификатор стола, за которым пользователь сидит в данный момент (или null если пользователь не в кафе). Тип int32. V = 4b

total_cost - текущая стоимость посещения заведения в рублях. Тип int32. V = 4b

tariff_id - выбранный пользователем тариф. Тип int32. V = 4b

start_time - время начала посещения кафе. Тип String. V = 12b

end_time - время ухода из кафе. Тип String. V = 12b

Таблица “food”

food_id - уникальный идентификатор пункта меню. Тип int32. V = 4b

name - название пункта меню. Тип String. V = 100b

content - состав. Тип String. V = 1000b

status - статус пункта меню (доступен/недоступен). Тип boolean. V = 1b

Таблица “orders”

order_id - уникальный идентификатор заказа еды. Тип int32. V = 4b

food_id - уникальный идентификатор пункта меню. Тип int32. V = 4b

status_done - статус приготовления . Тип boolean. V = 1b

employee_id - уникальный идентификатор работника принявшего заказ. Тип int32. V = 4b

table_id - уникальный идентификатор стола, за которым сидит заказавший посетитель. Тип int32. V = 4b

Таблица “bookings”

booking_id - уникальный идентификатор брони стола. Тип int32. V = 4b

table_id - номер стола. Тип int32. V = 4b

date - дата. Тип String. V = 12b

time - время начала сессии. Тип String. V = 12b

tariff_id - тариф, который выбрал пользователь, бронировавший стол, для себя. Тип int32. V = 4b

Таблица “employee”

employee_id - уникальный идентификатор работника. Тип int32. V = 4b

password - пароль от аккаунта. Тип String. V = 30b

contacts - электронная почта. Тип String. V = 50b

employment_date - дата найма на работу. Тип String. V = 14b

status - статус работника (работает/в отпуске/уволен). Тип String. V = 12b

permission_id - роль в заведении. Тип int32. V = 4b

Таблица “games”

game_id - уникальный идентификатор игры. Тип int32. V = 4b

name - название игры. Тип String. V = 30b

players_min - минимальное количество игроков. Тип int16. V = 2b

players_max - максимальное количество игроков. Тип int16. V = 2b

table_id - номер стола за которым сейчас игра. Тип int32. V = 4b

Таблица “tariff”

tariff_id - уникальный идентификатор тарифа. Тип int32. V = 4b

name - название тарифа. Тип String. V = 30b

cost_per_min - стоимость минуты в рублях. Тип int16. V = 2b

foods_id - id пунктов меню входящих в тариф. Тип String. V = 4 * n. n - количество пунктов меню ~ 40. V = 320b

Таблица “permission”

permission_id - уникальный идентификатор роли. Тип int32. V = 4b

name - название роли. Тип String. V = 30b

3.2.3. **Оценка удельного объема информации, хранимой в модели**

Для расчетов возьмем 1 месяц работы кафе, считая что средний посетитель посещает заведение 1.3 раза за тот срок.

Для четверых пользователей потребуется в среднем:

Три записи в таблице “guests”

Одна запись в таблице “booking”

Шесть записей в таблице “Orders”

Для одного работника потребуется в среднем:

Одна запись в таблице “employee”

Полставки = 20(часов в неделю) * 4(недели) * 2 (заказа в час положим средним значением) = 160 записей в таблице “orders” в среднем

Объем данных для хранения Nf пунктов меню, Ngm игр, Nt столов, Ntf тарифов, Ng гостей, Ne работников и Np ролей:

V(Nf, Ngm, Nt, Ntf, Ng, Ne, Np) = Nf * Vf + Ngm * Vgm + Nt * Vt + Ntf * Vtf + Ng * Vg + Ng * 1.5 * Vo + Ng * 0.25 * Vb + Ne * Ve + Np * Vp

Итого при объемах равных соответственно Vf = 1105, Vgm = 42, Vt = 94, Vtf = 356, Vg = 98, Vo = 21, Vb = 40, Ve = 114, Vp = 34 в среднем на месяц требуется:

V(Nf = 30, Ngm = 40, Nt = 12, Ntf = 1, Ng = 100, Ne = 5, Np = 2) = 30 * 1105 + 40 * 42 + 12 * 94 + 1 * 356 + 100 * 98 + 5 * 114 + 2 * 34 + 5 * 114 + 2 * 34 + 100 * 1.5 * 21 + 100 * 0.25 * 40 = 51540 b ~ 50.3 Kb

Объем базы в зависимости от количества пользователей: V(Ng) = 30 * 1105 + 40 * 42 + 12 * 94 + 1 * 356 + Ng * 98 + 5 * 114 + 2 * 34 + 5 * 114 + 2 * 34 + Ng * 1.5 * 21 + Ng * 0.25 * 40 = 37590 + 139,5Ng b

3.2.4. **Запросы к модели, с помощью которых реализуются сценарии использования**

Для реализации основного сценария гостя:

Поиск пользователя в коллекции Guests

SELECT * from Guests WHERE login = login AND password = password

Поиск свободных столов на дату 2023-09-20

SELECT table_number from Tables WHERE table_id NOT IN (SELECT table_id FROM Bookings WHERE date = “2023-09-20”)

Заказ столика на дату 2023-09-20 8:00

INSERT INTO Bookings (booking_id , table_id , date , time , tariff_id, guest_id) VALUES (booking_id, selectedTableID, selectedDate, :"8:00", selectedTariffID, guest_id)

3.3. Сравнение моделей

3.3.1. При увеличении количества объектов каждой сущности до разумных пределов наибольшее влияние на вес и реляционной, и нереляционной модели будет оказывать коллекции сущностей Столы и Посетители.

3.3.2. Обе модели избыточны по памяти для экономии по времени. Избыточными данными являются само наличие полей брони в сущности Стол и полей резервации в сущности Посетитель.

3.3.3. Запросы к реляционной и нереляционной моделям одинаковы по количеству запросов для совершения юзеркейса в зависимости от числа объектов в бд и прочих параметров и количеству задействованных коллекции. А именно для совершения юзеркейса задействуются 4 коллекции: 2 раза Посетителей, N раз Столов (в зависимости от их занятости), по 1 разу Тарифов и Брони.

3.3.4. Для оценки удельного объема информации сравним объемы баз в зависимости от количества пользователей для хранения Nf пунктов меню, Ngm игр, Nt столов, Ntf тарифов, Ng гостей, Ne работников и Np ролей, при условии Nf = 10, Ngm = 20, Nt = 10, Ntf = 2, Ng = 30, Ne = 3, Np = 2 :

- Для нереляционной БД: 5,48 + 0.34 Ng Kb
- Для реляционной БД: V(Ng) = 13547 + 132.5Ng b = 13.23 + 0,12 Ng Kb

Итого, если количество посетителей будет превышать 58 (из равенства формул), Реляционная модель будет выгоднее по памяти. Это будет сильнее ощущаться, если в среднем каждый день в антикафе будет приходить более 10 человек, что вполне реально.

3.3.5. Вывод.

Подводя итоги, реляционная модель данных будет более удачным выбором с точки зрения памяти и простоты модели для популярного антикафе с 60 и более посетителями в месяц.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Приложение было разработано на фреймворке Nuxt.js, работающем на Node.js. Для реализации бэкенд-логики использовались API маршруты Nuxt.js, в то время как для фронтенда использовался Vue.js.

4.2. Использованные технологии

База данных:

- MongoDB

Back-end:

- Node.js
- Nuxt.js API Routes

Front-end:

- Vue.js
- Nuxt.js

4.3. Снимки экрана приложения

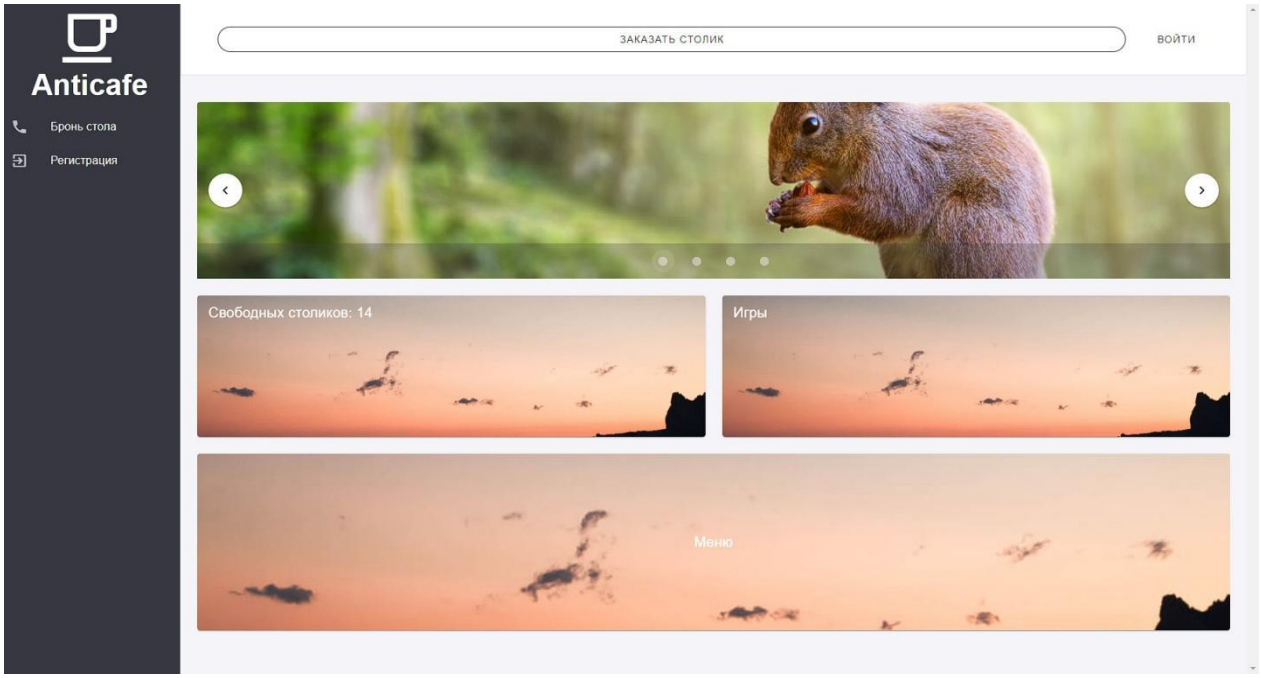

Рисунок 7 - Главный экран

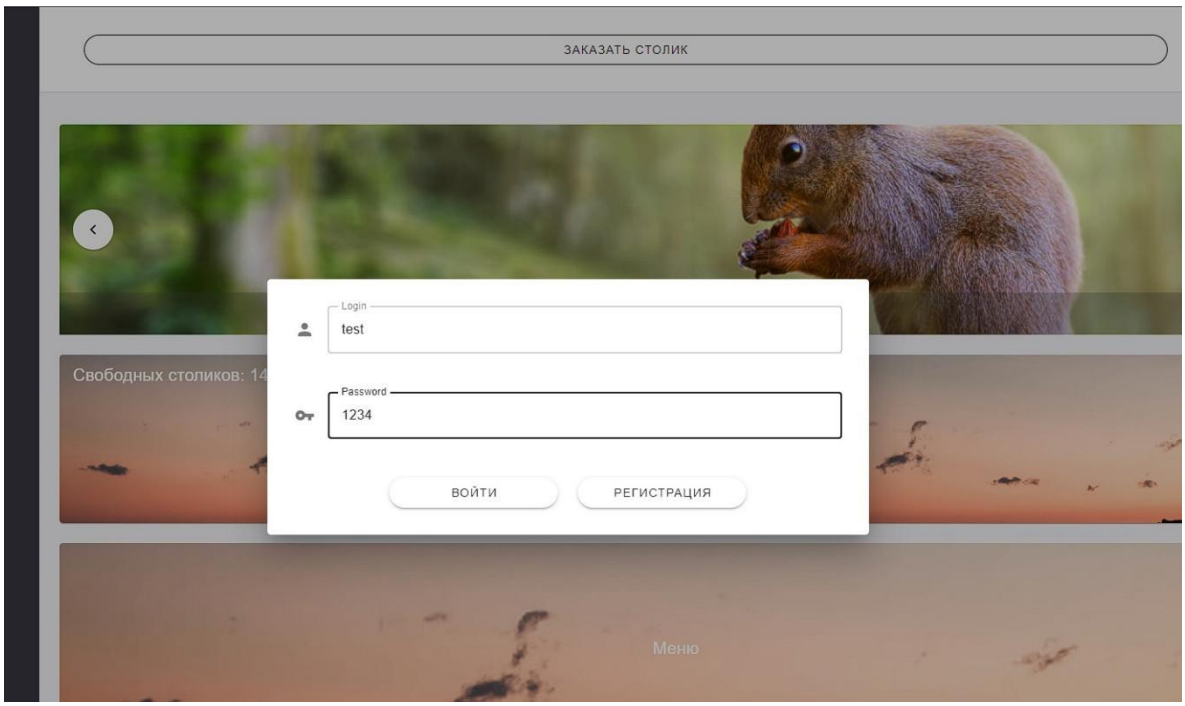

Рисунок 8 - Вход в аккаунт

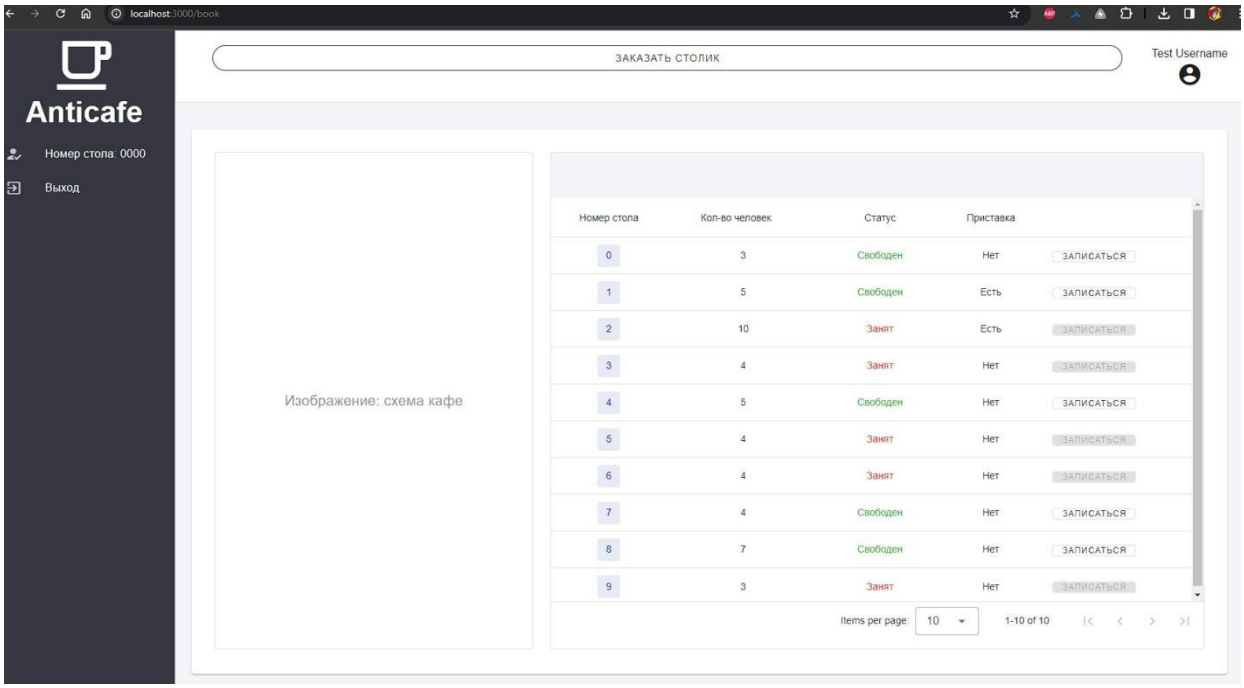

Рисунок 9 - Таблица столиков

5. ВЫВОДЫ

5.1. На текущем этапе развития приложения, мы успешно реализовали бэкенд, авторизацию, некоторые страницы фронтенда, на которых можно в виде таблиц просматривать содержимое базы данны.

5.2. Недостатки и пути для улучшения полученного решения

В данный момент, наше приложение реализовано не полностью и требует значительных доработок во всех сферах. Например, необходимо добавление всех страниц и связей между ними, реализация фильтров.

5.3. Будущее развитие решения

В перспективе планируется расширение функционала приложения, включая постоянную работу с большим количеством пользователей и объемом данных, а также выполнение полного функционала антикафе.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория
2. Запустить команду: docker compose up

6.2. Инструкция для пользователя

1. Открыть приложение: http://localhost:3000/

7. ЛИТЕРАТУРА

1. Документация к Neo4j: https://www.mongodb.com/docs/
2. Официальная документация Nuxt: https://nuxt.com/docs/
3. Официальная документация Vue: https://ru.vuejs.org/v2/guide/